

Bilinear Modeling Solution Approach for Fixed Charge Network Flow Problems*

Steffen Rebennack[†]

Artyom Nahapetyan[‡]

Panos M. Pardalos[†]

February 2, 2009

Abstract

We present a continuous, bilinear formulation for the fixed charge network flow problem. This formulation is used to derive an exact algorithm for the fixed charge network flow problem converging in a finite number of steps. Some preliminary computational experiments are reported to show the performance of the algorithm.

Key words: Bilinear modeling; fixed charge network flow problem; exact formulation; concave minimization

1 Introduction

Fixed charge network flow problems have a wide spectrum of applications. Among them are problems in network design, scheduling, production planning, supply chain and transportation science [13]. The fixed charge network flow problem is \mathcal{NP} -hard [14]; while, for instance, the single source uncapacitated case with a fixed number of fixed arc cost is strongly polynomial solvable [31].

Exact solution methods for the fixed charge network flow problem often utilize a binary mixed integer programming formulation together with Branch & Bound [11, 4, 24, 15], Branch & Cut

*Steffen Rebennack and Panos M. Pardalos are partially supported by NSF and AirForce grants. The support is greatly appreciated.

[†]Department of Industrial & Systems Engineering, Center for Applied Optimization, University of Florida, Gainesville, FL 32611, USA, {steffen.pardalos@ufl.edu}

[‡]Innovative Scheduling Inc., Gainesville Technology Enterprise Center (GTEC), 2153 SE Hawthorne Road, Gainesville, FL 32641, artyom@innovativescheduling.com

[23], Benders decomposition [6], dynamic programming [12] or exploit the fact that the optimum in a concave minimization problem is obtained in a vertex [20]. There are a huge variety of heuristic methods [10, 16, 18, 17, 8] which are especially favorable for large-scale problems due to the \mathcal{NP} -hardness.

In this paper, we present a continuous, bilinear formulation for the fixed charge network flow problem with a concave cost leading to an exact solution algorithm. The algorithm is based on earlier work by NAHAPETYAN and PARDALOS [22, 21].

In Section 2, we discuss a linear and a bilinear semi-continuous formulation for the fixed charge network flow problem. A continuous bilinear model is then presented in Section 3. Computational feasibility tests are made in Section 4. We conclude with Section 5.

2 Two Mathematical Formulations of the Fixed Charge Network Flow Problem

In this section, we discuss two formulations of the fixed charge network flow problem over a continuous domain. The first formulation has a semi-continuous objective function with constraints describing the network topology. The second formulation has the same feasible region but the objective is defined in terms of piecewise linear functions. In this section, we show that the second formulation is equivalent to the first one under certain conditions.

Let $G = (V, A)$ represent a directed network with node set V and arc set A . Let us assume that the network has $n = |V|$ nodes and $m = |A|$ arcs. With each arc $a \in A$, we associate a capacity $\lambda_a \in \mathbb{Z}_+$, and $\lambda \in \mathbb{Z}_+^{|A|}$ denotes the corresponding vector of capacities. Let matrix B represent the $n \times m$ node-arc incidence matrix of network G [7], and b_i denote the demand ($b_i \leq 0$) or the supply ($b_i \geq 0$) at node $i \in V$ with $\sum_{i \in V} b_i = 0$, [1].

In the fixed charge network flow problem, there is a variable cost of $c_a \in \mathbb{Z}_+$ and a fixed cost of $s_a \in \mathbb{Z}_+$. This gives the following (concave) cost structure for each arc $a \in A$

$$f_a(x_a) = \begin{cases} s_a + c_a x_a, & \text{if } x_a \in (0, \lambda_a] \\ 0, & \text{if } x_a = 0 \end{cases} . \quad (1)$$

Using above described notations, we can state the following formulation for the fixed charge network flow problem [21]:

$$\begin{aligned} FCNF : \quad & \min_x & f(x) &= \sum_{a \in A} f_a(x_a) \\ & \text{s.t.} & B\mathbf{x} &= \mathbf{b} \\ & & 0 &\leq \mathbf{x} \leq \lambda \end{aligned}$$

Let us approximate the concave cost function f_a in (1) by a piece-wise linear concave function

$$\phi_a^{\epsilon_a}(x_a) = \begin{cases} s_a + c_a x_a, & \text{if } x_a \in [\epsilon_a, \lambda_a] \\ c_a^{\epsilon_a} x_a, & \text{if } x_a \in [0, \epsilon_a] \end{cases} \quad (2)$$

with $c_a^{\epsilon_a} = c_a + \frac{s_a}{\epsilon_a}$, [21]. Recognize that $\phi_a^{\epsilon_a}$ underestimates function f_a for all $\epsilon_a > 0$, see Figure 1.

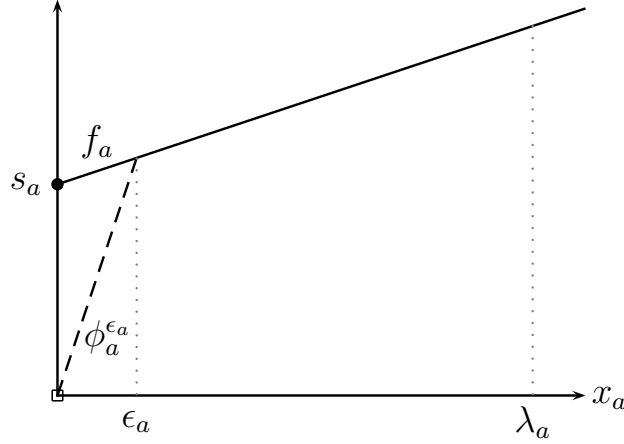


Figure 1: Approximation of function $f_a(x_a)$

In analogon to *FCNF* formulation, we can define the continuous piece-wise linear network flow problem for a given $\epsilon > 0$ as

$$\begin{aligned} CPLNF(\epsilon) : \quad & \min_x \quad \phi^\epsilon(x) = \sum_{a \in A} \phi_a^{\epsilon_a}(x_a) \\ & \text{s.t.} \quad B\mathbf{x} = \mathbf{b} \\ & \quad \mathbf{0} \leq \mathbf{x} \leq \lambda \end{aligned}$$

In [21], authors proved that for a sufficiently small ϵ , $CPLNF(\epsilon)$ correctly models the fixed charge network flow problem. Below we briefly restate the theorem and for the proof we refer to [21].

Let \mathcal{V} represent the set of vertices of the feasible region of *FCNF*. Then, define δ as the minimum among all positive components of all vertices x^v in \mathcal{V} ; *i.e.* $\delta = \min \{x_a^v \mid x^v \in \mathcal{V}, a \in A, x_a^v > 0\}$.

Theorem 2.1. [21] *Let \mathbf{x}^* (\mathbf{x}^ϵ) be an optimal solution to *FCNF* (*CPLNF*(ϵ)). Then, $\phi^\epsilon(\mathbf{x}^\epsilon) = f(\mathbf{x}^*)$ for all ϵ with $\epsilon_a \in (0, \delta) \forall a \in A$.*

3 Bilinear Formulation

In this section, we develop a continuous bilinear formulation for the fixed charge network flow problem, which is equivalent to $CPLNF(\epsilon)$ formulation. Based on the new formulation, we describe an algorithm that can be used to find an optimal solution to the fixed charge network flow problem.

Note that $CPLNF(\epsilon)$ has linear constraints and a piecewise linear continuous concave objective function. Next we modify the objective function using binary variables, indicating in which one of the two parts the function lies. Let us define binary variables y_a for all arcs $a \in A$

$$y_a = \begin{cases} 0, & \text{if } x_a \in [0, \epsilon_a) \\ 1, & \text{if } x_a \in (\epsilon_a, \lambda_a] \end{cases} . \quad (3)$$

Relaxing these variables allows us to formulate the following continuous bilinear network flow problem for any given $\epsilon > 0$

$$CBLNF(\epsilon) : \quad \varphi(\epsilon) = \min_{x,y} \quad \sum_{a \in A} \left(c_a^{\epsilon_a} x_a + (s_a - \frac{s_a}{\epsilon_a} x_a) y_a \right) \quad (4)$$

$$\text{s.t.} \quad B\mathbf{x} = \mathbf{b} \quad (5)$$

$$\mathbf{0} \leq \mathbf{x} \leq \lambda \quad (6)$$

$$\mathbf{0} \leq \mathbf{y} \leq \mathbf{1} \quad (7)$$

In the following theorem, we prove that $CBLNF(\epsilon)$ and $CPLNF(\epsilon)$ are equivalent.

Theorem 3.1. $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal solution to $CBLNF(\epsilon)$, if and only if \mathbf{x}^* is optimal for $CPLNF(\epsilon)$.

Proof. To prove the theorem, below we show that any optimal solution of one of the problems corresponds to a feasible solution for the second problem with the same objective function value.

“ \Leftarrow ” Let $\bar{\mathbf{x}}$ denote an optimal solution of $CPLNF(\epsilon)$. Assign variable $\bar{\mathbf{y}}$ values according to (3). Then, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is feasible to $CBLNF(\epsilon)$ and has the same objective function value.

“ \Rightarrow ” Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ denote an optimal solution of $CBLNF(\epsilon)$. Obviously, $\bar{\mathbf{x}}$ is also feasible for $CPLNF(\epsilon)$. Next, we need to prove that the objective function values are equal. Doing so, we need to show that if $s_a \neq 0$ and $\bar{\mathbf{x}}_a \neq \epsilon$, then $\bar{\mathbf{y}}$ respects its definition (3). As we are only considering the effect of variable $\bar{\mathbf{y}}$, we can ignore the term $c_a^{\epsilon_a} \bar{\mathbf{x}}_a$ in the objective function (4). The remainder can be restated as

$$\bar{\mathbf{y}}_a \left(s_a - \frac{s_a}{\epsilon_a} \bar{\mathbf{x}}_a \right) .$$

Case 1: Let $\bar{x}_a < \epsilon$. Then $s_a - \frac{s_a}{\epsilon_a} \bar{x}_a > 0$. Since in $CBLNF(\epsilon)$ we minimize the objective function (4), at optimality \bar{y}_a has to be equal to 0.

Case 2: Let $\bar{x}_a > \epsilon$. Then $s_a - \frac{s_a}{\epsilon_a} \bar{x}_a < 0$. Similar to the previous case, at optimality \bar{y}_a has to be equal to 1. \square

Let us adopt Theorem 2.1 to the new model $CBLNF(\epsilon)$. This will be used as a stopping criteria for the algorithm introduced below.

Corollary 3.2 (Optimality Condition). *Let $(\mathbf{x}^*, \mathbf{y}^*)$ be the optimal solution of problem $CBLNF(\epsilon)$. If*

$$x_a^* \in \{0\} \cup [\epsilon, \lambda] \quad \forall a \in A \quad , \quad (8)$$

then \mathbf{x}^ solves the fixed charge network flow problem optimally.*

Proof. It follows from Theorem 2.1, as $\epsilon_1 \leq \epsilon_2$ implies that $\varphi(\epsilon_1) \geq \varphi(\epsilon_2)$. \square

Let x^* denote an optimal solution to the $FCNF$ and $f = f(x^*)$ represent its objective function value. We can make the following observations:

- The objective function of $CBLNF(\epsilon)$ is concave for any $\epsilon > 0$, as its Hessian is negative (semi)definit. Hence, $CBLNF(\epsilon)$ is \mathcal{NP} -hard to solve [26].
- $\varphi(\epsilon)$ is a lower bound on f for all $\epsilon > 0$.
- Any feasible solution $\hat{\mathbf{x}}$ to $CBLNF(\epsilon)$ is also feasible to the fixed charge network flow problem; however, the objective function value may differ. Hence, by evaluating $f(\hat{\mathbf{x}})$, we obtain also an upper bound on f .

With the optimality conditions (8) on hand, we can state an exact (continuous) algorithm for the fixed charge network flow problem; along the lines of Procedure 1 in [21]. This algorithm is given in Algorithm 3.1. Basically, a series of bilinear problems (Step 2) is solved to (global) optimality until the stopping criteria is met (Step 6).

Corollary 3.3. *The CBL Algorithm solves the fixed charge network flow problem in a finite number of iterations.*

Proof. The correctness of the algorithm follows from Theorem 3.1 together with Corollary 3.2. In the worst case, ϵ has to be updated in Step 4 of the CBL Algorithm until $\epsilon_a \leq \delta$ for each $a \in A$. Hence, for each arc $a \in A$, an upper bound $L = l + 1$ on the number of iterations is given by (the smallest) $l \in \mathbb{Z}_+$ such that $\beta^l \alpha \lambda \leq \delta$; it is $L + 1$ as $\alpha \lambda$ might be already less than or equal to δ and

Algorithm 3.1 Continuous BiLinear (CBL) Algorithm**Input:** Matrix B , vectors \mathbf{a}, \mathbf{b} , λ ; parameters $\alpha, \beta \in (0, 1)$ **Output:** Optimal \mathbf{x} and parameter ϵ

```

1: // Initialize
    $\epsilon = \alpha \cdot \lambda$ 
2: // Solve continuous bilinear problem
   solve  $CBLNF(\epsilon)$ ; let  $\bar{\mathbf{x}}$  be its optimal solution
3: // Check stopping criteria (8)
   if  $\exists \bar{x}_a \in (0, \epsilon)$  then
4:   // Update  $\epsilon_a$ 
      $\epsilon_a \leftarrow \beta \cdot \epsilon_a$ 
5:   // Start from the beginning with an updated  $\epsilon$ 
     GoTo Step 2
   else
6:   // Optimal solution found
     return  $\bar{\mathbf{x}}$  and  $\epsilon$ 

```

one iteration is needed to check the stopping criteria. An upper bound on the number of iterations is then given by

$$1 + \sum_{a \in A} \max \left\{ \left\lceil \log_{\beta} \left(\frac{\delta}{\alpha \lambda_a} \right) \right\rceil, 0 \right\} .$$

□

4 Computational Tests

In this section, we discuss some computational tests with the *CBL* Algorithm, developed in Section 3. The aim of these computational tests is to validate the proposed algorithm and to see its performance with respect to the convergence rate. For this purposes, it is sufficient to use a general (global) nonlinear solver rather than a specialized global solver for minimizing concave linearly constrained quadratic programs.

Let us briefly review some global algorithms for nonconvex quadratic programming. A finite Branch & Bound algorithm using semidefinite programming is given by BURER and VANDEN-BUSSCHE, [3]. A review of global optimization approaches for indefinite quadratic programming is given by PARDALOS [25] whereas a general overview with applications is given by FLOUDAS and VISWESWARAN [9]. Specialized algorithms for concave, linear constrained quadratic programming have been developed by KONNO [19], BOMZE and DANNINGER [2] as well as CHINCHULUUN et al. [5]. A parallel algorithm was introduced by PHILLIPS and ROSEN [27].

In order to reduce the number of iterations of the Continuous Bilinear Algorithm 3.1, in step 2

we use a heuristic called ‘‘Adaptive Dynamic Cost Updating Procedure (ADCUP)’’ described in [21]. Its basic idea is to iteratively fix \mathbf{y} or \mathbf{x} variables in formulation $CBLNF(\epsilon)$ and solve the resulting linear programs (LP) by dynamically updating the value of ϵ until a feasible solution is reached. ADCUP is a very fast converging algorithm and usually provides a feasible solution with objective function value very close to the optimal one. Next, we employ the feasible solution as a starting point and solve $CBLNF(\epsilon)$ via the Branch-And-Reduce Optimization Navigator (BARON) version 8.1.1 of SAHINIDIS et al. [28, 30, 29]. Algorithm 3.1 is modeled in the ‘‘General Algebraic Modeling System’’ (GAMS) version 22.6. The computational framework is Redhat version 5, running on a Dell power edge 2600 with two Pentium 4 3.2Ghz with 1MB cache processors and 6 GB of memory. For comparability purposes, only one processor is used for the tests.

For the computational tests, the data for the network is uniformly random generated. For each arc $a \in A$, we have the variable cost $c_a \in [1, 5]$ and fixed cost $s_a \in [50, 100]$. The supply and demand $|b_i| \in [30, 50]$ for all nodes $i \in V$. All tested graphs have two supply and two demand nodes. The upper bound on the flow is set to (two times) the total demand; *i.e.* $\lambda = \sum_{i \in V} |b_i|$. Parameters, α and β are set to 0.2 and 0.3, respectively.

Table 1: Computational results for different sizes of the networks. Shown are the results for 10 randomly generated instances per problem size

G		# inner iter.				# outer iter.			
n	m	min	max	average	σ	min	max	average	σ
15	100	6	34	17.0	11.32	1	4	1.4	0.97
15	200	15	86	45.2	21.01	1	17	4.1	5.86
20	100	10	95	29.1	25.40	1	4	1.5	0.97
20	200	2	104	51.6	32.27	1	1	1.0	0.00
25	100	7 ¹	38 ¹	21.2 ¹	12.74 ¹	1 ¹	7 ¹	1.9 ¹	1.96 ¹
25	200	2	124	40.6	33.84	1	23	3.8	6.86
30	100	2	27	13.0	9.18	1	2	1.4	0.44
30	200	2	30	13.1	9.31	1	6	1.5	1.58
35	100	4 ²	20 ²	10.2 ²	6.11 ²	1 ²	1 ²	1.0 ²	0.00 ²
35	200	2	71	26.2	20.30	1	13	3.5	4.06

¹ 1 instance is infeasible

² 4 instances are infeasible

For each network size, 10 randomly generated instances are solved and computational results are presented in Table 1. The number of inner iterations (# inner iter.) is the number of LP’s solved by ACUP heuristic procedure while the number of outer iterations (# outer iter.) reports how often $CBLNF(\epsilon)$ is solved until the global optimal is reached. The number of outer iterations is crucial for the performance of the algorithm as the computation of the global optimum of $CBLNF(\epsilon)$ is the bottleneck of the CBL Algorithm. The average number of outer iterations is quite low, ranging from 1.0 to 3.8. There are only 8 instances out of 100 tested cases where the number of outer iterations is larger than 4. Typically, the heuristics is called only once or twice after the first bilinear model has been solved. Hence, the largest fraction of the inner iterations takes the first call

of the heuristics. Typically, after $CBLNF(\epsilon)$ is solved and an updated ϵ is passed to the heuristics along with the currently best solution, only two iterations are needed during the heuristics, *i.e.* two times \mathbf{y} variables have to be updated after the first LP has been solved.

In order to check if the global optimum is reached after termination of the algorithm, we solve a mixed-integer programming formulation for the fixed charge network flow problem with ILOG CPLEX. The objective function values (and even the solutions) are the same for all tested instances. However, we have to mention that CPLEX requires only a very small fraction of the CPU time used by the CBL Algorithm to solve the tested instances.

The CBL Algorithm has two parameters α and β to tune. Parameter α is not very crucial for the convergence, as it effects only the initial value of ϵ . However, parameter β has a significant impact on the inner as well as the outer number of iterations. This effect is shown in Figure 2. As expected, if β is closer to one, then more inner as well as outer iterations are needed; for value 0.99, 1023 LP's and 205 bilinear models have to be solved. In contrary, if the value is closer to 0, then less iterations are required. However, for the tested instances, there was no difference in the number of inner as well as outer iterations for the values 0.01, 0.001, 0.0001 and 0.00000001. This is also expected, as at some point, each ϵ_a is updated only once during the algorithm. Nevertheless, choosing parameter β too small leads to numerical stability problems, as the cost in model $CBLNF(\epsilon)$ depends on $\frac{1}{\epsilon}$.

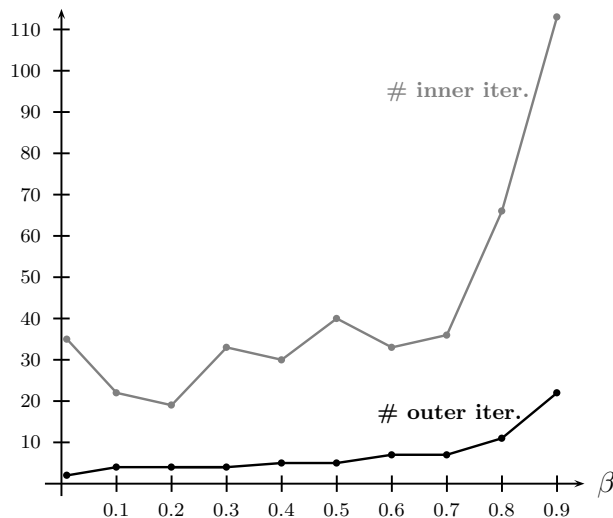


Figure 2: Computational tests for different values of parameter β for a graph with 15 nodes and 200 arcs

5 Conclusion

We discussed a continuous bilinear model formulation with network flow constraints for the fixed charge network flow problem. This formulation lead to an exact algorithm which converges in a finite number of steps. Computational feasibility is shown by various numerical test.

References

- [1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [2] I. M. Bomze and G. Danninger. A Global Optimization Algorithm for Concave Quadratic Programming Problems. *SIAM Journal on Optimization*, 3(4):826–842, 1993.
- [3] S. Burer and D. Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming*, 113:259–282, 2008.
- [4] A. Cabot and S. Erenguc. Some branch-and-bound procedures for fixed-cost transportation problems. *Nav. Res. Log. Q.*, 31:145–154, 1984.
- [5] A. Chinchuluun, P. M. Pardalos, and R. Enkhbat. Global minimization algorithms for concave quadratic programming problems. *Optimization*, 54(6):627–639, 2005.
- [6] Alysson M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429–1450, June 2005.
- [7] R. Diestel. *Graph Theory*. Springer, New-York, Electronic Edition 2000.
- [8] B. Eksioglu, S. Duni-Eksioglu, and Panos. M. Pardalos. *Equilibrium Problems and Variational Models*, chapter Solving Large Scale Fixed Charge Network Flow Problems. Kluwer Academic Publishers, 2001. A. Maugeri, F. Giannesi, and P. M. Pardalos (eds.).
- [9] C. Floudas and V. Visweswaran. *Handbook of Global Optimization*, chapter Quadratic optimization, pages 217–269. Kluwer Academic Publishers, 1995. R. Horst and P. Pardalos (eds.).
- [10] D. B. M. M. Fontes and J. Goncalves. Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50(1):67–76, 2007.
- [11] Dalila B. Fontes, Eleni Hadjiconstantinou, and Nicos Christofides. A branch-and-bound algorithm for concave network flow problems. *Journal of Global Optimization*, 34(1):127–155, January 2006.
- [12] Dalila B.M.M. Fontes, Eleni Hadjiconstantinou, and Nicos Christofides. A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. *European Journal of Operational Research*, 174(2):1205–1219, October 2006.
- [13] J. Guenes and P. M. Pardalos. *Supply Chain Optimization*. Springer, 2005.
- [14] G. M. Guisewite and P. M. Pardalos. Minimum Concave-Cost Network Flow Problems: Applications, Complexity, and Algorithms. *Annals of Operations Research*, 25:75–100, 1990.
- [15] W. Hirsch and G. Dantzig. The fixed charge problem. *Nav. Res. Log. Q.*, 15:413–424, 1968.
- [16] D. Kim, X. Pan, and P. M. Pardalos. An Enhanced Dynamic Slope Scaling Procedure with Tabu Scheme for Fixed Charge Network Flow Problems. *Computational Economics*, 2–3:273–293, 2006.

-
- [17] D. Kim and P. M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24:195–203, 1999.
- [18] D. Kim and P. M. Pardalos. Dynamic Slope Scaling and Trust Interval Techniques for Solving Concave Piecewise Linear Network Flow Problems. *Networks*, 35(3):216–222, 2000.
- [19] H. Konno. A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11:14–27, 1976.
- [20] K. Murty. Solving the fixed charge problem by ranking the extreme points. *Oper. Res.*, 16:268–279, 1968.
- [21] A. Nahapetyan and P. M. Pardalos. Adaptive Dynamic Cost Updating Procedure for Solving Fixed Charge Network Flow Problems. *Computational Optimization and Applications*, 39:37–50, 2008.
- [22] Artyom Nahapetyan and Panos M. Pardalos. A Bilinear Relaxation Based Algorithm for Concave Piecewise Linear Network Flow Problem. *Journal of Industrial and Management Optimization*, 3:71–85, 2007.
- [23] F. Ortega and L. A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.
- [24] U. Palekar, M. Karwan, and S. Zionts. A branch-and-bound method for fixed charge transportation problem. *Manag. Sci.*, 36:1092–1105, 1990.
- [25] P. M. Pardalos. Global optimization algorithms for linearly constrained indefinite quadratic problems. *Computers & Mathematics with Applications*, 21(6-7):87–97, 1991.
- [26] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is np-hard. *Journal of Global Optimization*, 1:15–22, 1991.
- [27] A. T. Phillips and J. B. Rosen. A parallel algorithm for constrained concave quadratic global minimization. *Mathematical Programming*, 42(1-4):421–448, 1988.
- [28] H. S. Ryoo and N. V. Sahinidis. Global optimization of non-convex NLPs and MINLPs with application in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.
- [29] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, March 1996.
- [30] N. V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, March 1996.
- [31] H. Tuy. Strong Polynomial Time Solvability of Minimum Concave Cost Network Flow Problem. *ACTA MATHEMATICA VIETNAMICA*, 25(2):209–218, 2000.